# 开发工具环境布局

开发工具：1、Adnroid-Adt-bundle 2、Android Studio

Adnroid-Adt-bundle 谷歌已经不支持下载sdk

ADT Bundle包含了Eclipse、ADT插件和SDK Tools，是已经集成好的IDE，只需安装好Jdk即可开始开发，推荐初学者下载ADT Bundle，不用再折腾开发环境

配置SDK环境步骤：

第一步：新建名称ANDROID\_HOME 值C:\adt-bundle-windows-x86\_64-20140702\sdk

第二步：修改Path,添加%ANDROID\_HOME%\tools;%ANDROID\_HOME%\build-tools;%ANDROID\_HOME%\platform-tools

第三步：打开DOS窗口，检查是否配置成功：输入 adb显示adb版本及命令帮助

输入android弹出窗口，成功

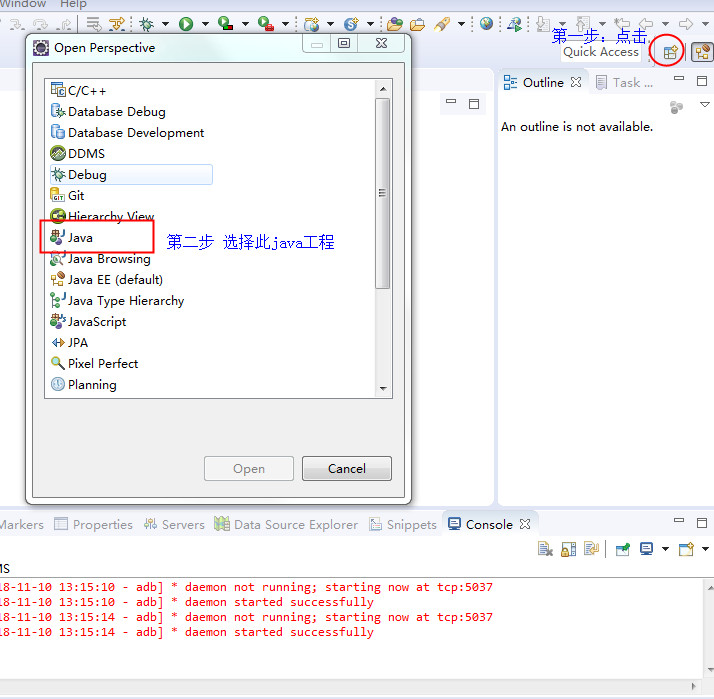
使用eclipse搭建环境

搭建eclipse环境(需要ADT的zip包)

第一步 下载ADT包：最新的好像是ADT-23.0.7，进行部署最新eclipse(2018年9月份)

配置离线ADT：Eclipse主窗口Help —> install New soft…. —> Add—> Archive —> 找到zip文件ADT-23.0.7 —>进行Add安装

第二步 安装好后：选择java工程才有安卓开发目录



第三步 eclipse选择SDK目录(系统配置SDK环境会自动选择)

window -🡪preferfences —>Android —>Browse.. —>找到自己的SDK目录即可

第四步 开发

右键 —>new —> AndroidApplicationProject

# android项目结构

src 我们编写的java代码。

gen （G）自动生成资源文件对应的代码

Android4.4

Android Private Libraries 安卓导入的依赖包

assets 资产目录 打包一些文件到应用程序里面（数据库文件，网页文件，配置文件）

bin 将java代码和资源编译打包成.apk文件

libs 安卓开发所需要存放的第三方开发包

res 资源目录

drawable 图标

layout 布局文件

menu 菜单配置文件

values

dimens 安卓的控件尺寸

strings 存储所有文本

styles 样式文件

colors 颜色文件

ids 存储安卓控件的id的文件

AndroidManifest.xml 安卓配置文件 相当于javaweb的properties文件

project.propertise 工程的配置文件 配置安卓版本 第三方依赖包的路径

proguard-project.txt 配置混淆编译的规则

# 应用程序编译安装过程

第一步.把所有的.java文件编译成.class文件

>jdk的javac.exe

第二步.把所有的.class字节码文件打包合成一个文件classes.dex(压缩文件)

>\sdk\build-tools\android-4.4目录下的dx.bat 工具

第三步.根据资源id,生成一个resources.ap\_ 保存所有的资源信息

第四步.把.dex文件、res目录资源文件、resources.ap\_文件 asset目录下的文件合成打包成一个apk压缩包

> 使用\sdk\build-tools\android-4.4\aapt.exe

第五步. adb.exe 负责把生成好的apk文件上传安装到手机里面。

> Android debug bridge 安卓调试桥 负责上传安装apk

# adb命令

1. 安装一个应用程序

adb install xxx.apk

2. 卸载一个应用程序

adb uninstall com.xx.xx.xx

3. 进入到手机终端（安卓手机是linux系统，此后就可以开始执行linux终端命令）

adb shell (ls -l查看当前目录下的文件 pwd查看当前文件路径 cd xxx 跳到某个文件夹下 cd.. 返回上一层)

4. 杀死adb的调试桥(当调试桥不稳定的时候) 和启动adb的调试桥

adb kill-server 杀死

adb start-server 启动

5. 查看连接到电脑上的设备

adb devices

6. 指定设备执行的adb指令

adb -s emulator-5556 install xxx.apk

# UI布局控件

View是所有控件的父类

通用属性：5种

1. 控件的宽高：（固定值 与两种状态）

android:layout\_width="30dp"

android:layout\_height="30dp"

2. 控件的可见度：（3种状态）

android:visibility="visible"

3. 控件的背景：（颜色 图片）

android:background="#F00" //RGB(红绿蓝): #000~#FFF; ARGB(透明度+红绿蓝): #0000~#FFFF

4. 盒子模型：

android:layout\_margin="10dp"

android:padding="5dp"

5. 点击事件：

android:onClick="haha"

## 布局(layout)

View与ViewGroup的关系：

1. View表示控件 View与View之间是无法嵌套

2. 为了让控件能够嵌套其他控件 才产生了ViewGroup

3.继承关系； ViewGroup一般比较少使用 我们一般使用其子类实现（也就是5大布局控件）

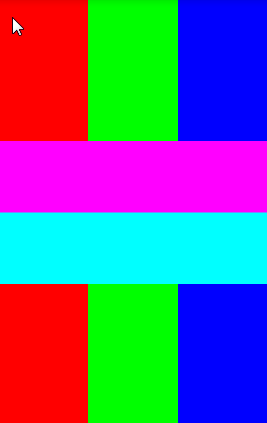
ViewGroup的子类5大布局控件：

1、LinearLayout: 线性布局指的是该布局下包含的子布局列表为横线或者竖线排布。

指定方向：(横向布局/纵向布局) android:orientation="" 值为：vertica和horizontal

权重：android:layout\_weight="1" 指的是布局中：1比1形式，有三个布局这个属性把父容量分3比例

效果图



2、RelativeLayout: 相对布局

1、如果什么也不做子控件就在容器的左上角

两种方式：1、相对父控件布局2 相对兄弟控件布局

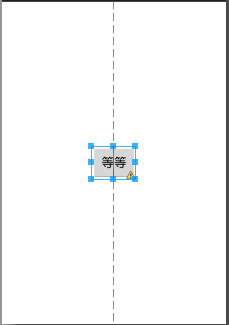
第一种方式：相对于父控件

android:layout\_centerHorizontal="true" 横向居中

android:layout\_centerVertical="true" 纵向居中

android:layout\_centerInParent ="true" 横向纵向居中 等同于使用了上面两个同时使用的效果

效果图



和父控件进行对齐

android:layout\_alignParentTop="true" 头顶上对齐

android:layout\_alignParentBottom="true" 底部对齐

android:layout\_alignParentLeft="true" 左边对齐

android:layout\_alignParentRight="true" 右边对齐

第二种方式：针对已存在的兄弟控件(在某个控件的上面/下面/左边/右边)

android:layout\_above ="@id/button1" 在id为button1控件的上边(但是不会对齐)

android:layout\_below ="@id/button1" 在控件的下边(但是不会对齐)

android:layout\_toLeftOf ="@id/button1" 在控件的左边(但是不会对齐)

android:layout\_toRightOf ="@id/button1" 在控件的右边(但是不会对齐)

对齐相对兄弟控件的边对齐

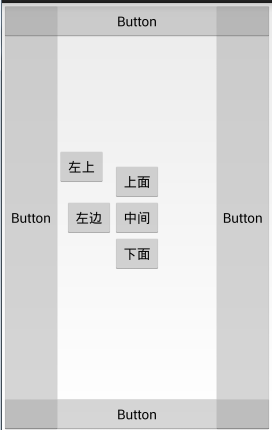
android:layout\_alignTop ="@id/button1" 本控件头部边和button1控件头部边对齐

android:layout\_alignBottom ="@id/button1" 本控件低部边和button1控件底部边对齐

android:layout\_alignLeft ="@id/button1" 本控件左边和button1控件左边对齐

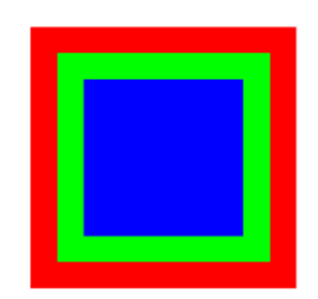
android:layout\_alignRight ="@id/button1" 本控件右边和button1控件右边对齐

效果图



3、FrameLayout: 帧布局 代码越是后面控件越在顶层 (应用场景: 霓虹灯) 层层叠加

效果图



4 AbsoluteLayout :绝对布局 控件在父控件中的绝对定位 (应用场景: 机顶盒开发)

android:layout\_x="30dp"

android:layout\_y="30dp

5、TableLayout: 表格布局 (应用场景: 银行表格) LinearLayout(线性布局)的子类

也不怎么用，效果丑：一般都线性布局来代替

## 文本、按钮和输入框

TextView是所有文本框的父类; TextView继承View

共同属性

android:text="文本"

android:textSize="20sp" 文字大小

android:textColor="#FF0" 文字颜色

android:textStyle="bold" bold加粗文字\italic斜体文字

android:lines="3" 只显示3行内容(没有此属性会全部显示出来)：针对属性text的内容

android:ellipsize="end" end尾部/ 属性text的内容太多以…来显示

android:singleLine="true" 变成单行而且内容太多以…来显示

android:typeface="monospace" 设置字体风格。字形有：normal, sans, serif,monospace

android:clickable=””

### Button

Button是TextView子类所以：属性与TextView基本相似。

不同点：

1. 按钮是自带了背景的控件

2. 按钮是可以点击了

优化button按钮操作 按钮有两种状态：按下和原来状态

第一种方式 引入两张图片：按下、原来 (引入操作看下面)

注意：默认(原来)的状态一定要写到最后面

在res文件夹中创建一个drawable文件夹：xml类型Drawable选择：selector：

<selector xmlns:android="http://schemas.android.com/apk/res/android" >

<!-- 第一步 设置按下的状态：引入自己定义的按下图片-->

state\_pressed="true"：意思是重定义系统按下后的状态，这里也可以定义别的状态：如选中的状态

android:drawable：引入自己放在每个不同分辨率下的图片

<item android:state\_pressed="true" android:drawable="@drawable/ic\_launcher"/>

<!--第二步 设置原来的状态 注意：原来状态只引入图片即可；默认(原来)的状态一定要写到最后面 -->

<item android:drawable="@drawable/ic\_launcher"/>

</selector>

第二种方式：改变背景原色操作 (引入操作看下面)

注意：默认(原来)的状态一定要写到最后面

在res文件夹中创建一个drawable文件夹：xml类型Drawable选择：selector：

<selector xmlns:android="http://schemas.android.com/apk/res/android" >

<!-- 第0步把两张图片放入drawable-hdpi下 -->

<!-- 第一步 设置按下的状态 先item 标签然后使用shape标签自己画-->

<item android:state\_pressed="true">

<shape >

<stroke android:color="#888" android:width="1dp"/> stroke 边框 color颜色 width边框粗细

<solid android:color="#FF0"/> solid （背景颜色）填充的颜色

<corners android:radius="10dp"/> corners 设置圆角

</shape>

</item>

<!--第二步 设置默认(原来)的状态 注意:默认的状态一定要写到最后面 -->

<item>

<shape >

<!--stroke边框 width:粗细 -->

<stroke android:color="#888" android:width="1dp"/>

<!-- 输入框内填充的颜色 -->

<solid android:color="#FFF"/>

<!-- 设置圆角 -->

<corners android:radius="10dp"/>

</shape>

</item>

</selector>

第三种方式 ：文字颜色改变的操作 (引入操作看下面)

<selector xmlns:android="http://schemas.android.com/apk/res/android" >

<!-- button效果 有两种状态：1 原来状态，2.按下的状态 -->

<!-- 第一步 设置按下的状态 (pressed加压)属性 -->

<item android:state\_pressed="true" android:color="#00F"/> 和前两种方式这里变成了color

<!--注意默认的状态一定要写到最后面 第二步 设置原来的状态 -->

<item android:color="#F00"/>

</selector>

上面三种方式引入操作：第一种和第二种以背景原色引入图片的方式引入，第三种文本颜色引入

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="2dp"

android:textColor="@drawable/btn\_text\_selector" //第三种方式的引入：引入字体颜色的状态

android:text="按钮"

android:background="@drawable/btn\_selector" /> //第一种和第二种：引入方式

#### 单选框

需要用到单选框的组

RadioGroup：继承布局控件的LinearLayout(线性布局)：可以用线性布局的纵向和横向

RadioButton：继承Button所以可以用text显示内容

获取单选框内容操作

第一步 布局xml

<RadioGroup

android:id="@+id/radio\_a"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="horizontal"> //因为RadioGroup继承了LinearLayout(线性布局)所以有此属性

<RadioButton

android:id="@+id/rb\_1" //需要id，使用事件CheckedChanged要

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:checked="true" 默认选中

android:text="男"/>

<RadioButton

android:id="@+id/rb\_2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="女"/>

</RadioGroup>

第二步 代码

第一步 获取RadioGroup的id

RadioGroup byId = (RadioGroup) findViewById(R.id.radio\_a);

第二步 使用事件ChangeListener

byId.setOnCheckedChangeListener(new OnCheckedChangeListener() {

@Override

//group 代表RadioGroup，checkedId选中的id

public void onCheckedChanged(RadioGroup group, int checkedId) {

//判断RadioButton中的id

if(checkedId==R.id.rb\_1) {

Toast.makeText(MainActivity.this, "你选择的是男的", Toast.LENGTH\_SHORT).show();

}else if(checkedId==R.id.rb\_2){

Toast.makeText(MainActivity.this, "你选择的是女的", Toast.LENGTH\_SHORT).show();

}

}

});

#### 多选框

CheckBox ：继承Button

第一步 布局 这里就只写一个

<CheckBox

android:id="@+id/cb\_1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="南瓜"

/>

第二步代码

CheckBox idc = (CheckBox) findViewById(R.id.cb\_1); //注意红字的，虽然和单选框的监听类名字一样但是不同的类

idc.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

//第一个看继承，第二个 布尔值

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

Toast.makeText(MainActivity.this, isChecked?"我选中了南瓜":"取消", Toast.LENGTH\_SHORT).show();

}

});

图标优化：

第一步 在res文件夹中创建一个drawable文件夹：然后创建文件xml类型Drawable选择：selector

<selector xmlns:android="http://schemas.android.com/apk/res/android" >

第一步 选中状态操作

<item android:state\_checked="true" state\_checked设置为true覆盖系统选中样式

android:drawable="@drawable/more\_radio\_selected"/>

第二步 默认(未选中)的必须要最后写：

<item

android:drawable="@drawable/more\_radio\_normal"/>

</selector>

第二步 引入：

<!-- CheckBox里面的选择框样式是由属性android:button决定的 -->

<CheckBox

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:button="@drawable/cbx\_slector"//引入操作

android:text="苹果" />

### EditText

EditText输入框继承TextView

属性

android:hint="请输入QQ" 提示语

android:editable="true" 允许编辑

android:maxLength="50" 设置最大的字数

android:inputType="textPassword" 密码框

优化EditText：设置为圆弧的输入框

第一步 在res文件夹中创建一个drawable文件夹：然后创建文件xml类型Drawable选择：shape：

<shape xmlns:android="http://schemas.android.com/apk/res/android" >

<!--stroke边框 color：颜色 width:粗细 -->

<stroke android:color="#888" android:width="1dp"/>

<!-- 输入框内填充的颜色 -->

<solid android:color="#FFF"/>

<!--设置边框的圆角-->

<corners android:radius="10dp"/>

</shape>

第二步 当作上面的就是一个背景图片来引入：

<EditText

android:layout\_width="match\_"

android:layout\_height="32dp"

android:layout\_margin="2dp"

android:paddingLeft="5dp" 设置光标的位置

android:background="@drawable/et\_shap"/> //以背景引入图片操作

效果图：C:\Users\Administrator\Desktop\捕获.PNG

## WebView

WebView是AbsoluteLayout (绝对布局)的子类

小案例：让网页在此控件显示

需要注意上网权限问题和旋转重新加载页面两个问题

上网权限问题：在AndroidManifest.xml文件中最后添加：

<uses-permission android:name="android.permission.INTERNET"/> 上网权限问题

旋转重新加载页：在AndroidManifest.xml文件中标签<activity>中添加属性

android:configChanges="orientation|keyboardHidden|screenSize"。 旋转问题

上网权限问题：看第三步

//第一步 获取控件：view所有控件的父类强转

WebView wv = (WebView) findViewById(R.id.webView1);

//第二步 获得控件后进行加载页面

wv.loadUrl("http://106.12.106.21:8080/");

//第三步 此时是没有联网的权限需在AndroidManifest.xml加权限

// 这句话不在这里写：<uses-permission android:name="android.permission.INTERNET"/>

//第四步 启动js代码 ：默认是不支持JavaScript的

WebSettings settings = wv.getSettings();

settings.setJavaScriptEnabled(true);

//第五步 设置重定向的网页也在此控件显示 ： 一旦重定向跳转网页会导致跳转到系统的浏览器来显示，

wv.setWebViewClient(new WebViewClient(){

//返回true 意思是：不要让系统来操作帮忙跳转网页

//view:自己写的WebView控件，url：每次一点击都会获得新的网页地址传入

public boolean shouldOverrideUrlLoading(WebView view, String url) {

view.loadUrl(url);

return true;

}

});

}

AndroidManifest.xml文件的内容

<uses-sdk

android:minSdkVersion="23"

android:targetSdkVersion="23" />

<!-- 上网权限问题 -->

<uses-permission android:name="android.permission.INTERNET"/>

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name=".Bill"

android:label="@string/app\_name"

<!-- 设置为旋转页面不重新加载 -->

android:configChanges="orientation|keyboardHidden|screenSize"

>

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

## 滑动控件

ScrollView是FrameLayout:(帧布局)的子类(上下滑动)

ScrollView：如属性text内容多手机屏幕无法一次性显示完可以手滑向下显示内容

注意：只能嵌套一个子控件，如果2个子控件会报错，可以嵌套一个容器如LinearLayout(线性布局)再嵌套其它控件就可以；如下：

<ScrollView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<LinearLayout 只能嵌套一个子类：为了能多个控件享受滑动，这里嵌套了容器控件

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="vertical"> 经测试必须是纵向才显示

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="50sp"

android:text="你好啊你好啊你好啊你好啊你好啊你好啊你好啊啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊你好啊"

/>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="50sp"

android:text="笑死我了啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊啊"

/>

</LinearLayout>

</ScrollView>

HorizontalScrollView(左右滑动)

注意事项和上下滑动一样

报错内容

java.lang.IllegalStateException: HorizontalScrollView can host only one direct child

HorizontalScrollView 或ScrollView内部只能持有一个直接的子元素

顾名思义，错误出现的原因即是ScrollView中有多个元素

解决办法：

把所有的子元素放到一个LinearLayout内部或RelativeLayout等其他布局方式后，再在这个layout外部用scrollview包住。

## 图片控件

ImageView直接继承View

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:scaleType="fitEnd" 图片如何展示

android:src="@drawable/ic\_launcher" 图片地方

/>

1. 设置图片源

android:src="@drawable/ic\_launcher" src属性只能放res/drawable下的图片

1. 代码设置图片源

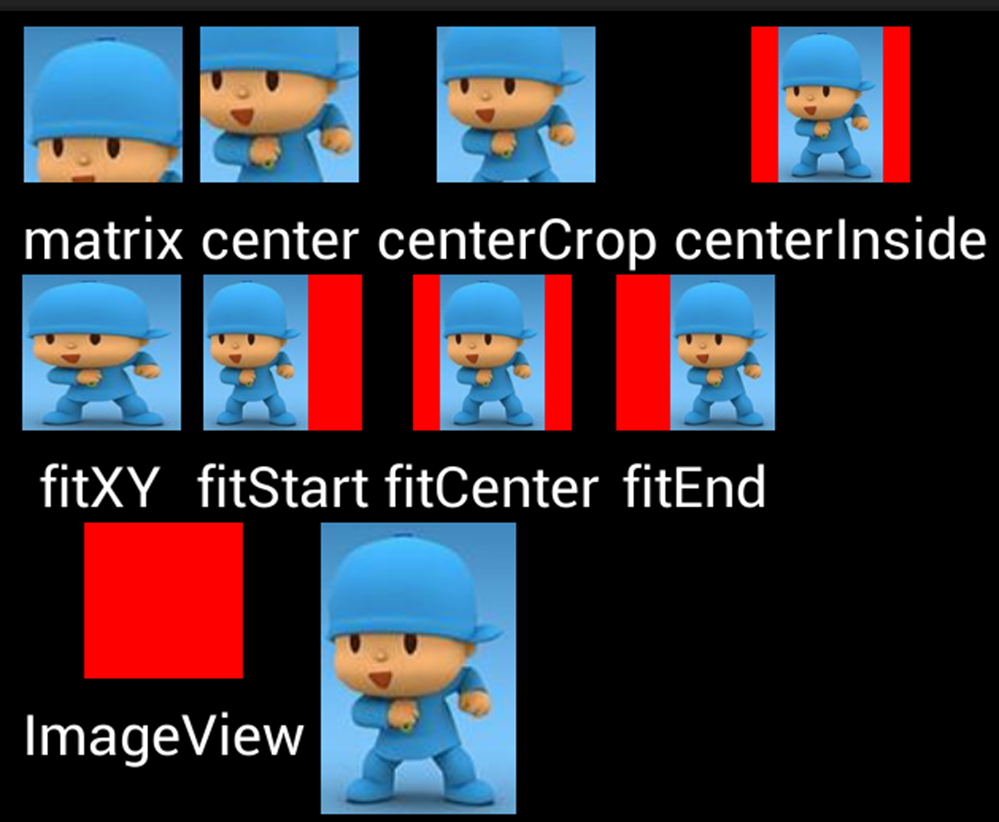
imageView.setImageResource(resId)

imageView.setImageBitmap(bm);

imageView.setImageDrawable(drawable);

3、scaleType属性的样式效果图

android:scaleType="" 下面图片是属性值和效果图



使用开源SmartImageView控件步骤：

第一步 导包：把android-smart-image-view-master包中的代码从com源码导入工程中

它有7个类：其中SmartImageView是控件类继承ImageView

第二步 主布局中activity\_main.xml使用SmartImageView自定义控件

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

//其它属性都和系统的ImageView一样

<com.loopj.android.image.SmartImageView

android:id="@+id/siv"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"/>

</RelativeLayout>

第三步 主代码中使用：其实都差不多，只是增强了系统的ImageView功能

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//找到控件SmartImageView

SmartImageView iv = (SmartImageView) findViewById(R.id.siv);

//这里增强方法：参数一：网络上取资源(注意网络权限)，参数二：取不到找默认图片。还有很多功能自行琢磨

iv.setImageUrl("http://192.168.0.100:8080/img/a.png", R.drawable.ic\_launcher); }

## 日期和时间选择器

DatePicker是FrameLayout:(帧布局)的子类

日期选择器: DatePicker

第0步：自己拖控件

//第一步获取控件

DatePicker datePicker = (DatePicker) findViewById(R.id.datePicker1);

//第一个年份，第二个月份(取值0~11)，第三个日期，第三个监听器可以为null

//datePicker.init(1986, 0, 19, null); 不需要时间可以为null

//获取内容需要监听器：更改日期自动打印时间

datePicker.init(1986, 0, 19, new OnDateChangedListener() {

@Override

public void onDateChanged(DatePicker view, int year, int monthOfYear, int dayOfMonth) {

String message=year+"年"+(monthOfYear+1)+"月份"+dayOfMonth+"日";

Toast.makeText(MainActivity.this, message, 0).show();

}

});

TimePicker是FrameLayout:(帧布局)的子类

时间选择器: 24小时制

第一步 获取控件

TimePicker timePicker = (TimePicker) findViewById(R.id.timePicker1);

//设置成24小时表示法：(控件显示就不会有上午和下午)

timePicker.setIs24HourView(true);

//设置时间改变监听器获取内容

timePicker.setOnTimeChangedListener(new OnTimeChangedListener() {

@Override

public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {

String message=hourOfDay+"小时"+minute+"分";

Toast.makeText(MainActivity.this, message, 0).show();

}

});

## 进度条

ProgressBar直接继承View

ProgressBar进度条的使用

<ProgressBar

android:id="@+id/progressBar3"

style="?android:attr/progressBarStyleHorizontal" 设置此属性是带进度的进度条样式

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="500" max 进度条的总长度

android:progress="125"/> 默认一开始的位置

SeekBar (用户拖动)可拖动的(普通)进度条 一般用来验证 SeekBar是ProgressBar孙子类

<SeekBar

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="500" max 进度条的总长度

android:progress="125" 默认一开始的位置

/>

RatingBar (用户拖动)可拖动的(星星)进度条 一般用来验证 RatingBar是ProgressBar孙子类

<RatingBar

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:numStars="5"

android:rating="3"

android:stepSize="0.3"

/>

进度条美观优化：

第一步 在res文件夹中创建一个drawable文件夹：xml类型Drawable选择：layer-list：

<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >

背景图操作

<item android:drawable="@drawable/progress\_bar\_bg"

android:id="@android:id/background"/> @android:id/background就是想告诉系统此图片为背景图

进度图操作

<item android:drawable="@drawable/progress\_bar\_selected\_bg" //自己的图片

android:id="@android:id/progress"/> @android:id/progress 告诉系统此图片为进度条图片

</layer-list>

第二步 引入

<!-- 引用上面的进度条样式图 -->

<ProgressBar

style="?android:attr/progressBarStyleHorizontal" 引入系统样式，可以按住ctrl后点击进入查看

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:progressDrawable="@drawable/pb\_bg" 这里进行引入

android:max="100"

android:progress="100" />

## 列表

### ListView

ListView也可以说属于布局中的：它是ViewGroup的儿子AdapterView适配器下的类

#### BaseAdapter

简单的布局操作

1、不关注性能 2、只有一个布局，一个控件

第一步 布局xml

布局:1、在layout 文件夹中创建LinearLayout布局取名tt.xml

<?xml version="1.0" encoding="utf-8"?> 注意：本身ListView就自带滑动功能所以不用嵌套ScrollView控件

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<ListView

android:id="@+id/lv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

设置背景图片

android:background="@drawable/ic\_launcher"

设置的背景图片之后：如果出现上拉或下拉背景图片不见了设置为透明，版本6.0测试没事

android:cacheColorHint="#0FF0"

分割条样式，不需要刻意设置为透明

android:divider="#FF0"

分隔条粗细设置

android:dividerHeight="1dp"

上拉到最顶后出现阴影进行设置

android:fadingEdge="none"

点击的时候会出现系统给的颜色，如果先去掉刻意设置为透明或者添加图片

android:listSelector="@drawable/ic\_launcher"/></LinearLayout>

第二步 MainActivity中代码

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.tt); //1、获取布局中的tt.xml

ListView lv = (ListView) findViewById(R.id.lv); //2、获取控件ListView

List<String> list = new ArrayList<String>(); //3、准备数据

for (int i = 0; i < 150; i++) {

list.add("测试"+i);

}

MyAdapter ma = new MyAdapter(list,this); //4、 因数据不是直接交互的需：适配器作为中间来处理数据

lv.setAdapter(ma);

}

第三步 MyAdapter适配器类

public class MyAdapter extends BaseAdapter { 1、继承BaseAdapter适配器

private List<String> list;

private Context c;

public MyAdapter(List<String> list,Context c) { 2、构造方法需要Comtext和数据

this.list=list;

this.c = c;

}

public int getCount() { 3、此方法每次进入此类系统都会调用：用来获取数据的长度

return list.size();设置长度

}

\* position :返回当前标签的索引

\* convertView : 缓存View：老师测试：刚进来加载显示:为null，虚拟机显示‘测试0’遮住后就会有缓存测试0的数据

\* parent：有值的view：主要用来获取context

public View getView(int position, View convertView, ViewGroup parent) {4、此方法每次进入此类系统都会调用：用来获取View

TextView tv =new TextView(c); 4.1、代码创建一个TextView控件

4.2设置控件属性宽高：注意：LayoutParams是内部类很多控件都有，如果没有找他父类，你要看你的控件是放在那里的，tv是放在ListView中所以要用到ListView的(它没有只有他父类AbsListView有内部类LayoutParams)

tv.setLayoutParams(new AbsListView.LayoutParams(AbsListView.LayoutParams.MATCH\_PARENT,

AbsListView.LayoutParams.WRAP\_CONTENT));

tv.setText(list.get(position)); //设置tv控件属性txt中内容

tv.setTextSize(30);

TextView tv2 =new TextView(c);

return tv; 5、返回此TextView的View

}

public Object getItem(int position) { 6、此方法需要外界调用如：在点击事件

return null;

}

public long getItemId(int position) { 7、此方法和点击事件参数四相关联

return 0;

}

使用打气筒：布局.xml转换为View

第一步和第二步：和简单布局一致 这里不重复了

第三步 需要再次创建一个布局：在layout 文件夹中创建LinearLayout布局取名hh.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<TextView 第一个TextView

android:id="@+id/tv"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:textSize="50dp"

/>

<TextView 第二个TextView

android:id="@+id/tv2"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:textSize="10dp"

/>

</LinearLayout>

第四步MyAdapter适配器类

这里只写不同点：

注意：构造方法就不用传Context了

public View getView(int position, View convertView, ViewGroup parent) {

LayoutInflater li= LayoutInflater.from(parent.getContext()); 1获取打气筒类: 就是可以将一个布局转换成一个View

View view = li.inflate(R.layout.hh, null); 2、获取布局hh.xml，参数一：布局.xml,参数二：ViewGroup容器

TextView tv = (TextView) view.findViewById(R.id.tv); 3、获取hh.xml布局中的TextView控件

tv.setText(list.get(position)); 4、设置控件内容

TextView tv2 = (TextView) view.findViewById(R.id.tv2); 5、获得第二个控件

tv2.setText(list.get(position));

return view; 6、返回ciview

}

调用android系统中自带的TextView标签

//(1)布局充气的一个类 就是可以将一个布局转换成一个View

LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());

//(2)创建所有标签View的父类：参数1：系统自带的layout布局 2是否需要添加到某个容器里面去

convertView = layoutInflater.inflate(android.R.layout.simple\_list\_item\_2,null);

//(3)通过findViewById获取系统自带控件TextView

TextView txv = (TextView) convertView.findViewById(android.R.id.text1);//获取第一个TextView

TextView txv2 = (TextView) convertView.findViewById(android.R.id.text2);//获取第二个TextView

(4)返回View

return convertView;

性能的问题的改造

第一步 创建一个内部类用于保存ListView容器每一个子容器的控件

class ViewHolder{

TextView tv2; 控件1

TextView tv3; 控件2

}

第二步 方法getView的改造

在滑动的过程中不停的会调用getView

position :返回当前标签的索引

convertView : 缓存View：老师测试：刚进来加载显示:为null，虚拟机显示‘测试0’遮住后就会有缓存测试0的数据

parent：相当于ListView中的子容器,每次返回的view都会直接添加到此容器来：主要用来获取context

第一次进入此方法position会显示手机屏幕能显示的内容、convertView此时都为null，当进行滑动完第一个控件就会进行缓存

public View getView(int position, View convertView, ViewGroup parent) {

ViewHolder vh=null; 1、内部类创建

if(convertView ==null) { 2、第一次进入：没有滑动内容的时候convertView都为 null

vh=new ViewHolder(); 3、创建内部类对象

LayoutInflater li= LayoutInflater.from(parent.getContext()); 4、获取打气筒

View view = li.inflate(R.layout.hh, null); 5、获取View

convertView=view; 6、赋值到缓存view中

vh.tv2 = (TextView)convertView.findViewById(R.id.tv2); 7 、获取TextView控件并赋值到内部类成员变量中

vh.tv3= (TextView)convertView.findViewById(R.id.tv3);

convertView.setTag(vh); 8、每一个控件都有一个tag：参数是object

}else {

vh= (ViewHolder) convertView.getTag(); 9、获取内部类

}

vh.tv2.setText(list.get(position)); 10、获取第一次进入创建控件对象(已有别的属性，这里只是修改属性text内容)

vh.tv3.setText(list.get(position));

return convertView; 11、返回缓存view即可

}

}

#### 监听器

private MyAdapter adapter; // MyAdapter弄成成员变量用于调用getItem(int position)方法

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ListView lv = (ListView) findViewById(R.id.lv);

//准备数据

final List<String> list =new ArrayList<String>();

for (int i = 0; i < 30; i++) {

list.add("测试"+i);

}

adapter= new MyAdapter(list);

lv.setAdapter(adapter);

点击监听器OnItemClickListener

lv.setOnItemClickListener(new OnItemClickListener() {

//参数一；参数二：；参数三：当前索引；参数四：是MyAdapter中getItemId返回值(重要)

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

Toast.makeText(MainActivity.this, "你当前点击的是："+list.get(position), 0).show();

adapter.getItem(position);//MyAdapter类中的方法使用

}

});

滑动监听器setOnScrollListener

lv.setOnScrollListener(new OnScrollListener() {

//滑动的状态改变时候调用，参数一：本身ListView 参数二：使用OnScrollListener调用静态常量

public void onScrollStateChanged(AbsListView view, int scrollState) {

switch (scrollState) {

case OnScrollListener.SCROLL\_STATE\_FLING: //快速滑动 (清扫)

Toast.makeText(MainActivity.this, "清扫", 0).show();

break;

case OnScrollListener.SCROLL\_STATE\_IDLE: //不动的时候

Toast.makeText(MainActivity.this, "空闲", 0).show();

break;

case OnScrollListener.SCROLL\_STATE\_TOUCH\_SCROLL: //按住鼠标慢移动

Toast.makeText(MainActivity.this, scrollState==0?"0":"慢移动", 0).show();

break;

default:

break;

}

}

//滑动的时候被调用，参数二：屏幕显示最上面的第一个索引，参数三：屏幕显示控件总数，参数四：整个ListView的长度

public void onScroll(AbsListView view, int firstVisibleItem, int visibleItemCount, int totalItemCount) {

Log.v("wwl", "first="+firstVisibleItem+"visible="+visibleItemCount+"total="+totalItemCount);

}

});

#### SimpleAdapter

什么时候使用SimpleAdapter都可以

1、准备数据 2、准备好布局---找到布局下的子控件---->数据跟子控件进行绑定

第一步 两个布局：一个只有ListView，一个ImageView和TextView控件

ListView布局名称：activity\_main.xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<ListView

android:id="@+id/lv"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"></ListView>

</LinearLayout>

两个控件布局：用于嵌套到ListView列表控件中

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="horizontal" >

<ImageView

android:id="@+id/iv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_vertical"

android:background="#FFFF0000"/>

<TextView

android:id="@+id/tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_vertical"/>

</LinearLayout>

第二步 MainActivity主方法代码

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

ListView lv = (ListView) findViewById(R.id.lv);

//1、准备数据

List<Map<String, Object>> datas=new ArrayList<Map<String,Object>>();

for(int i=0;i<50;i++) {

Map<String,Object> map = new HashMap<String, Object>();

map.put("图片", R.drawable.ic\_launcher); //添加key：”图片”值：引入图片的id

map.put("文字", "测试"+i); //添加key：” 文字”值：引入内容

datas.add(map);

}

2、准备好布局---找到布局下的子控件---->数据跟子控件进行绑定

SimpleAdapter adapter=new SimpleAdapter(this, datas, R.layout.tt, new String[]{"图片","文字"}, new int[] {R.id.iv,R.id.tv});

lv.setAdapter(adapter);

}

#### ArrayAdapter

文字适配器 ：ArrayAdapter当一个列表里面只有显示一个TextView 就可以直接使用它

setContentView(R.layout.tt); 1、获取自己的布局(此布局看BaseAdapter一样的)

ListView lv = (ListView) findViewById(R.id.lv); 2、获取布局中的控件

List<String> list = new ArrayList<String>(); 3、准备数据：可用List集合或者数组

for (int i = 0; i < 150; i++) {

list.add("测试"+i);

}

4、使用ArrayAdapter适配器

参数二： 子项布局 这里使用了android的布局

参数三：将每一项数据绑定到哪个TextView下，使用了系统自带的布局就必须使用系统的TextView

参数四：数据

ArrayAdapter<String> a= new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, android.R.id.text1, list);

lv.setAdapter(a); 5、控件绑定适配器

## 网格

GridView和ListView一样，是兄弟类

第一步：布局:tt.xml和hh.xml 步骤几乎和List一样

tt.xml内容：使用LinearLayout只有一个控件GridView

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<GridView

android:id="@+id/lv"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:numColumns="5" 设置横向有几个控件

android:listSelector="#5F00"/> 设置点击背景颜色

</LinearLayout>

hh.xml 使用LinearLayout 子控件有ImageView和TextView

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical"

android:layout\_marginRight="5dp"

android:gravity="center">

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/ic\_launcher"/>

<TextView

android:id="@+id/tv3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="#5500FF00"

android:gravity="center"

android:textSize="10sp"/>

</LinearLayout>

第二步 MainActivity主方法代码

setContentView(R.layout.tt); 1、找到布局tt

GridView lv = (GridView) findViewById(R.id.lv); 2、获取网格控件

final List<String> list = new ArrayList<String>(); 3、准备数据

for (int i = 0; i < 150; i++) {

list.add("测试"+i);

}

MyAdapter ma = new MyAdapter(list); 4、使用适配器放入数据和控件交互

lv.setAdapter(ma);

lv.setOnItemClickListener(new OnItemClickListener() { 5、监听器和ListView一模一样的有两个

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

Toast.makeText(MainActivity.this, list.get(position), 0).show();

}

});

第三步 MyAdapter适配器类代码

public class MyAdapter extends BaseAdapter {

private List<String> list;

public MyAdapter(List<String> list) {

this.list=list;

}

public int getCount() {

return list.size();

}

public View getView(int position, View convertView, ViewGroup parent) {

//第一次进入此方法position只会显示手机屏幕能显示的内容、convertView此时都为null，当把一个控件滑动完就会进行缓存

ViewHolder vh=null;

if(convertView ==null) {

vh=new ViewHolder();

LayoutInflater li= LayoutInflater.from(parent.getContext());

View view = li.inflate(R.layout.hh, null);

convertView=view;

vh.tv3= (TextView)convertView.findViewById(R.id.tv3);

convertView.setTag(vh);

}else {

vh= (ViewHolder) convertView.getTag();

}

vh.tv3.setText(list.get(position));

return convertView;

}

class ViewHolder{

ImageView tv2;

TextView tv3;

}

public Object getItem(int position) { 此方法需要外界调用如：在点击事件

return null;

}

public long getItemId(int position) { 此方法和点击事件参数四相关联

return 0;

}

## 下拉框(了解)

Spinner不重要现实中不用这个控件来写

布局xml

activity\_main布局只有Spinner控件

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<Spinner

android:id="@+id/sp"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:spinnerMode="dialog" /> 还有别的属性

</RelativeLayout>

lv\_item\_layout 布局xml： TextView控件

<?xml version="1.0" encoding="utf-8"?>

<TextView xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="24sp"

android:textColor="#F00"

android:gravity="center" //内容中间显示

android:id="@+id/tv" />

MainActivity代码

setContentView(R.layout.activity\_main);

//1.有了一个列表

mSpinner = (Spinner) findViewById(R.id.sp);

//2.创建一些数据

ArrayList<String> datas=new ArrayList<String>();

for (int i = 0; i < 150; i++) {

datas.add("测试:"+i);

}

//3.将数据导入到ListView

MyAdapter adapter=new MyAdapter(datas);

mSpinner.setAdapter(adapter);

MyAdapter适配器代码

public class MyAdapter extends BaseAdapter {

private ArrayList<String> mDatas;

public MyAdapter(ArrayList<String> datas) {

mDatas=datas;

}

public int getCount() {

Log.v("520it", "getCount");

return mDatas.size();

}

public View getView(int position, View convertView, ViewGroup parent) {

// LayoutInflater 布局充气的一个类 就是可以将一个布局转换成一个View

LayoutInflater layoutInflater = LayoutInflater.from(parent.getContext());

// root 询问用户是否需要添加到某个容器里面去

View view = layoutInflater.inflate(R.layout.lv\_item\_layout, null);

TextView tv=(TextView) view.findViewById(R.id.tv);

tv.setText(mDatas.get(position));

return view;

}

## 折叠面板

ExpandableListView 上面三个监听器等都是一样的，这个不同

布局xml

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<ExpandableListView

android:id="@+id/explv"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</RelativeLayout>

主方法代码

private ExpandableListView mExpLv;

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

mExpLv=(ExpandableListView) findViewById(R.id.explv);

//1.组的数据

ArrayList<String> groupNames=new ArrayList<String>();

for (int i = 0; i < 3; i++) {

groupNames.add("组 "+i);

}

//2.成员的数据

HashMap<String, ArrayList<String>> childNames=

new HashMap<String, ArrayList<String>>();

for (int i = 0; i < groupNames.size(); i++) {

ArrayList<String> children=new ArrayList<String>();

for (int j = 0; j < 5; j++) {

children.add("成员"+j);

}

//遍历下组名 key就是组名 值就时一个成员队列

childNames.put(groupNames.get(i), children);

}

MyAdapter adapter=new MyAdapter(groupNames,childNames);

mExpLv.setAdapter(adapter);

//为ExpandableListView的组设置折叠监听器

// mExpLv.setOnGroupCollapseListener(onGroupCollapseListener)

// mExpLv.setOnGroupExpandListener(onGroupExpandListener)

// mExpLv.setOnGroupClickListener(onGroupClickListener)

// mExpLv.setOnChildClickListener(onChildClickListener)

mExpLv.setOnGroupExpandListener(new OnGroupExpandListener() {

@Override

public void onGroupExpand(int groupPosition) {

Toast.makeText(MainActivity.this, groupPosition+" 被展开了", 0).show();

}

});

}

MyAdapter适配器代码

public class MyAdapter extends BaseExpandableListAdapter { //继承另一个适配器类

private ArrayList<String> mGroupNames;

private HashMap<String, ArrayList<String>> mChildNames;

public MyAdapter(ArrayList<String> groupNames,

HashMap<String, ArrayList<String>> childNames) {

mGroupNames=groupNames;

mChildNames=childNames;

}

public int getGroupCount() {

return mGroupNames.size();

}

public View getGroupView(int groupPosition, boolean isExpanded,

View convertView, ViewGroup parent) {

LayoutInflater inflater = LayoutInflater.from(parent.getContext());

View itemView = inflater.inflate(android.R.layout.simple\_list\_item\_1, null);

TextView tv = (TextView) itemView.findViewById(android.R.id.text1);

tv.setText(mGroupNames.get(groupPosition));

return itemView;

}

public int getChildrenCount(int groupPosition) {

//1.获取组

String groupName = mGroupNames.get(groupPosition);

//2.根据组名获取子列表

ArrayList<String> childNames = mChildNames.get(groupName);

return childNames.size();

}

public View getChildView(int groupPosition, int childPosition,

boolean isLastChild, View convertView, ViewGroup parent) {

LayoutInflater inflater = LayoutInflater.from(parent.getContext());

View itemView = inflater.inflate(android.R.layout.simple\_list\_item\_1, null);

//1.控件已经被获取了

TextView tv = (TextView) itemView.findViewById(android.R.id.text1);

//2.获取子视图的数据

//3.获取组

String groupName = mGroupNames.get(groupPosition);

//4.根据组名获取子列表

ArrayList<String> childNames = mChildNames.get(groupName);

//5.从子列表里面获取某个Item

String text = childNames.get(childPosition);

tv.setText(text);

return itemView;

}

//下面方法老师没讲可有可无

@Override

public Object getGroup(int groupPosition) {

// TODO Auto-generated method stub

return null;

}

@Override

public Object getChild(int groupPosition, int childPosition) {

// TODO Auto-generated method stub

return null;

}

@Override

public long getGroupId(int groupPosition) {

// TODO Auto-generated method stub

return 0;

}

@Override

public long getChildId(int groupPosition, int childPosition) {

// TODO Auto-generated method stub

return 0;

}

@Override

public boolean hasStableIds() {

// TODO Auto-generated method stub

return false;

}

@Override

public boolean isChildSelectable(int groupPosition, int childPosition) {

// TODO Auto-generated method stub

return false;

}

}

## 自定义控件

第一步：写类继承控件，如ImageView

系统ImageView没有直接通过网络获取的资源的控件方法，这里继承加强ImageView控件

public class MyImageView extends ImageView{

第一步 构造方法的说明：

//只有参数Context(上下文)的时候：程序员在new一个控件的时候调用

public MyImageView(Context context) {

super(context);

}

两个参数：上下文和attrs：当把控件设置在xml文件中的时候 系统在渲染界面就会自动调用该构造器

public MyImageView(Context context, AttributeSet attrs) {

super(context, attrs);

}

下面代码是更改控件ui的用的：更换主线程操作(网络请求成功和失败)

private static final int SUCCESS=0x001;

private static final int FAILURE=0x002;

private Handler handler=new Handler() { //子线程更换主线程需要创建的类handler

public void handleMessage(Message msg) {

if(msg.what==SUCCESS) { //成功使用网络图片

setImageBitmap((Bitmap) msg.obj);

}else if(msg.what==FAILURE) { //失败使用默认图片

setImageAlpha(R.drawable.ic\_launcher);

}

};

};

第二步 ：自定义增强方法setImageUrl：网络请求图片资源直接设置到控件中

public void setImageUrl(final String urlpatch) {

new Thread() { 网络请求需要在子线程中执行

public void run() {

try {

URL url = new URL(urlpatch);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

int code = conn.getResponseCode();

if(code==200) {

//如果成功显示网路图片

InputStream is = conn.getInputStream();

Bitmap bitmap = BitmapFactory.decodeStream(is);

Message msg=new Message();

//msg.what 当有多个子线程向Handler发送信息Handler需要根据标示来进行处理

msg.obj=bitmap;

msg.what=SUCCESS;

handler.sendMessage(msg);

}else {

//失败显示默认图片(失败图片)

Message msg=new Message();

msg.what=FAILURE;

handler.sendMessage(msg);

}

} catch (Exception e) {

Message msg=new Message();

msg.what=FAILURE;

handler.sendMessage(msg);

e.printStackTrace();

}

}

}.start();

}

第二步 主布局xml添加自定义图片控件

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" >

<com.example.ui.MyImageView 这里更改全路径类名，其它属性和系统自带ImagView一样

android:id="@+id/iv"

android:layout\_width="80dp"

android:layout\_height="80dp"

android:layout\_marginTop="16dp"/>

<TextView

android:id="@+id/titleTV"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"/>

</RelativeLayout>

第三步 引用：其实一模一样的

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

MyImageView iv = (MyImageView) findViewById(R.id.iv);

iv .setImageUrl("http://192.168.0.100:8080/img/a.png");//这两步就会显示网络上的图片了

# android存储

## 获取assets文件夹

创建AssetManager资源的管理器

AssetManager assets = getAssets();

在assets目录下找到某个文件 将某个文件读进内存中

InputStream fis = assets.open("weather.xml");

## 内部存储

获取内部存储的目录：获取方法ENVironment.getDataDirectory()

File dataFile=ENVironment.getDataDirectory();

Log.v(“路径：”,dataFile.getAbsolutePath()) 打印：/data

前提测试：使用流直接在系统根目录是存储不了的，只有在/data/data/自己包名/目录下才能保存

String codeStr = code.getText().toString(); //获取EditText中输入的账号

String passwordStr = password.getText().toString(); //获取EditText中输入的密码

String string=codeStr+"#"+passwordStr

File file = new File("code.txt"); //错误：提示只有只读权限

File file = new File("/data/data/com.qq.wwl/code.txt"); 在自己的包名下可以保存

BufferedWriter bf =new BufferedWriter(new FileWriter(file));

bf.write(string);

bf.close();

上面保存麻烦；谷歌提供了：保存数据到文件中有两个目录files目录和cache目录

files目录：路径:/data/data/应用的包名/files两个方法：getFilesDir()和openFileOutput(name, mode)

清除此目录保存数据的操作：设置🡪 应用程序🡪应用详情🡪清除数据(删除cache目录 删除files目录)

**返回值File方法：getFilesDir() 保存的数据只能自己读写 （没有files目录自动创建目录）**

代码获取files目录方法： File file = new File(getFilesDir(), "/filelog.txt");

String string= codeStr+"#"+passwordStr

File file = new File(getCacheDir(),"code.txt"); 此时保存在了：/data/data/工程包名/files/code.txt

BufferedWriter bf =new BufferedWriter(new FileWriter(file));

bf.write();

bf.close();

**返回值是流：写出也会保存到files文件夹下 ，可以指定权限 （没有files目录自动创建目录）**

获取FileOutputStream流：openFileOutput(name, mode)可以指定文件读权限问题

获取FileInputStream流：openFileInput(name, mode)

使用openFileOutput(name, mode)把文件写出也会保存到files文件夹下

try {

//参数：第一个：文件名，第二个是：文件权限(参数看下面)

//这样即保存到了files文件中，文件权限：-rw-rw-r—别的用户就可读不可修改的权限

FileOutputStream fileOutput = openFileOutput("code.txt", Context.MODE\_WORLD\_READABLE);

//希望可读可写操作：进行”+”号操作或直接写7,文件权限变成-rw-rw-rw- 注意修改不了执行权限x，只能使用命令

// FileOutputStream t = openFileOutput("code.txt", Context.MODE\_WORLD\_READABLE+ Context.MODE\_WORLD\_READABLE);

//FileOutputStream t = openFileOutput("code.txt", 7);

BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fileOutput));

bw.write(string);

bw.close();

} catch (IOException e) {

e.printStackTrace();

}

第二个参数的具体说明

// openFileOutput：两个参数name 就是文件名，mode 文件权限

// MODE\_APPEND 比如一个文件已经有内容了 再次写文件就是在文件后面继续追加

// MODE\_PRIVATE 对其他应用私有化 但是本应用是可读可写的

// MODE\_WORLD\_READABLE 任何应用都能读取 代码会出现过时(其实是提示比较危险)

// MODE\_WORLD\_WRITEABLE 任何应用都能写入 代码会出现过时(其实是提示比较危险)

注意：

第一 此目录一般保存的是重要的信息 用户信息 应用的配置

第二 注意版本路径问题： 测试系统版本 4.1.2 和6.0 调用 Context.getFilesDir.getAbsolutePath 方法。

4.1.2 返回 /data/data/package/files

6.0 返回 /data/user/0/package/files)

6.0系统以后是 /data/user/x/package/files x 代表用户号 6.0 支持多用户

第三 权限问题：安卓系统一个应用一个用户，保存的到files目录中文件默认都是：-rw-------

cache目录：路径:/data/data/应用的包名/cache

此目录可以用来缓存不重要的数据 保存图片 等

清除此目录保存数据的操作：设置🡪 应用程序🡪 应用详情🡪 清除数据(删除cache目录下的文件)

代码获取cache目录方法：//getFilesDir() 使用方法： File file = new File(getCacheDir(), "文件名");

String string= codeStr+"#"+passwordStr

File file = new File(getCacheDir(),"code.txt"); 此时保存在了：/data/data/工程包名/cache/code.txt

BufferedWriter bf =new BufferedWriter(new FileWriter(file));

bf.write();

bf.close();

## 外部存储(SD卡)

代码：获取方法ENVironment.getDataDirectory() //调用此方法都能获取不同厂商路径目录

File dataFile=ENVironment.getDataDirectory();

Log.v(“路径：”,dataFile.getAbsolutePath()) //根据手机型号不同而不同,获取外部存储的目录

SD卡读写操作

1. 将文件写入到SD卡上首先要判断SD卡是否装载

//1.SD卡是可以插拔的 判断下SD卡是否装载

String externalState = Environment.getExternalStorageState(); //获取SD卡状态

if (externalState.equals(Environment.MEDIA\_MOUNTED)) { // 判断是否MOUNTED(安装)

File parentFile = Environment.getExternalStorageDirectory();

File file=new File(parentFile, "葫芦娃.avi");

//字节流

FileOutputStream fos=new FileOutputStream(file);

byte[] buff=new byte[1024\*1024];

for (int i = 0; i < 10; i++) {

fos.write(buff);

}

fos.close();

2、因为读写SD卡是敏感的行为 需要用户同意所以需要使用权限

找到AndroidMenifest.xml文件，在后面添加使用权限：

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

## SharedPreferences

SharedPreferences是一种轻型的Android数据存储方式，基于XML文件存储key-value数据

作用

通常用来存储一些简单的配置信息。如：软件更新要变量值，手势密码等

SharedPreferences对象本身只能获取数据而不支持存储和修改，存储修改是通过Editor对象实现。比较经典的使用方式例如用户输入框对过往登录账户的存储

存储位置

其存储位置在/data/data/<包名>/shared\_prefs目录下。

使用SharedPreferences保存key-value对的步骤如下：

（1）使用Activity类的getSharedPreferences方法获得SharedPreferences对象，其中存储key-value的文件的名称由getSharedPreferences方法的第一个参数指定，第二个参数指定访问应用程序私有文件的权限。

（2）使用SharedPreferences接口的edit获得SharedPreferences.Editor对象。

（3）通过SharedPreferences.Editor接口的putXxx方法保存key-value对。其中Xxx表示不同的数据类型。例如：字符串类型的value需要用putString方法。

（4）通过SharedPreferences.Editor接口的commit方法保存key-value对。commit方法相当于数据库事务中的提交（commit）操作。

a、打开Preferences，名称为setting，如果存在则打开它，否则创建新的Preferences

SharedPreferences userSettings = getSharedPreferences("setting", MODE\_PRIVATE); //参数：文件名，和files中流的第二个参数一样

b、让setting处于编辑状态

Editor editor = userSettings.edit();

c、存放数据

editor.putString("name"," abc#1234");

editor.putString("pwd","123");

d、完成提交

editor.commit();

打开保存的文件数据结构：

<map>

<string name=” name”>abc#1234</string>

<string name=” pwd”>123</string>

</map>

读取数据信息，步骤如下：

a、获取Preferences

SharedPreferences userSettings= getSharedPreferences("setting", MODE\_PRIVATE);

b、取出数据

String name = userSettings.getString("name","");参数二为空即可

String url = userSettings.getString("URL","");

删除和清空操作，如下：

a、清除指定数据

SharedPreferences.Editor editor = userSettings.edit();

editor.remove("KEY");

editor.commit();

b、清空数据

SharedPreferences.Editor editor = userSettings.edit();

editor.clear();

editor.commit();

注意：如果在 Fragment 中使用SharedPreferences 时，需要放在onAttach(Activity activity)里面进行SharedPreferences 的初始化，否则会报空指针 即 getActivity()会可能返回null ！

### XML序列化器生成文件和解析

xmlSerializer 创建xml文档的一个类 它可以帮助我们创建一个xml文件

public static void saveXmlByXmlSerializer(Context c, String passport, String password) throws Exception {

XmlSerializer xmlSerializer = android.util.Xml.newSerializer(); //使用安卓提供的Xml类构建序列化的XMl

// 1.告诉序列化工具 文件应该存储到什么地方

FileOutputStream fos = c.openFileOutput("qqinfo.xml",Context.MODE\_PRIVATE);

xmlSerializer.setOutput(fos, "utf-8");

// 2.告诉序列器 开始写文档首先创建xml的标头

xmlSerializer.startDocument("utf-8", true); // <?xml version='1.0' encoding='utf-8' standalone='yes' ?>

xmlSerializer.startTag(null, "map"); //参数一：命名空间，参数二 开始标签名称

xmlSerializer.startTag(null, "pwd"); //参数一：命名空间，参数二 开始标签名称

xmlSerializer.text(password); //为标签体中赋值

xmlSerializer.endTag(null, "pwd"); // pwd的闭标签

xmlSerializer.startTag(null, "username"); //参数一：命名空间，参数二 开始标签名称

xmlSerializer.text(passport); //为标签体中赋值

xmlSerializer.endTag(null, "username"); // username的闭标签

xmlSerializer.endTag(null, "map"); // map的闭标签

//3.文档写完 此时可以将内存中数据写到硬盘中去

xmlSerializer.endDocument();

4.最后写完结构：

<?xml version='1.0' encoding='utf-8' standalone='yes' ?>

<map>

<pwd>123</pwd>

<username>zhangsan</username>

</map>

安卓解析Xml不是dom和sax使用的是XmlPullParser它和sax差不多

public static String parseXmlFile(Context c) throws Exception {

String result="";

// 1.创建一个Pull解析器

XmlPullParser pullParser = Xml.newPullParser();

// 2.告诉解析器文件在哪里(将整个xml文件进行关联)

// 不管openFileInput/openFileOutput 都会在files目录下找文件

FileInputStream fis = c.openFileInput("qqinfo.xml");

pullParser.setInput(fis, "utf-8");

// 3.解析文件

// 3.1获取事件的类型 (开标签 文本 闭标签 标签属性 开始文档的类型 结束文档的类型)

int eventType = pullParser.getEventType();

//Log.v("520it", eventType + " ");

3.2pullParser.next() 返回的是下一个事件类型

//eventType = pullParser.next(); 打印2

3.3 pullParser.getName() 获取标签的名称

//Log.v("520it", pullParser.getName() + " "); //打印：map

3.4 如果是Text的事件 pullParser.getName()返回null 看源码：类型是4就是text事件

3.5 .nextText() 找到开始标签后面的值:<t>你好</t> 拿到：”你好”的值

使用循环解析

while (eventType!=XmlPullParser.END\_DOCUMENT) {

// <map> <pwd> <username>

if (eventType==XmlPullParser.START\_TAG) { //开标签 状态类型是2 闭标签状态类型是3

if (pullParser.getName().equals("username")) {

result+=(" username="+pullParser.nextText());//pullParser.nextText() 找到某个开标签后面的值

}else if (pullParser.getName().equals("pwd")) {

result+=(" pwd="+pullParser.nextText());

}

}

eventType = pullParser.next();

}

return result;

}

另一个列子：文件放在assets文件夹中：获取标签体内的天气文本内容

String result="";

//1.找到文件 AssetManager资源的管理器

AssetManager assets = getAssets();

//在assets目录下找到某个文件 将某个文件读进内存中

InputStream fis = assets.open("weather.xml");

//2.创建一个Pull解析器

XmlPullParser pullParser=Xml.newPullParser();

//3.绑定流

pullParser.setInput(fis, "utf-8");

//4.开始解析文档 事件解析-->获取一开始的事件类型

int eventType = pullParser.getEventType();

//5.做一个while循环 结束条件:直到读取到文档结束

while (eventType!=XmlPullParser.END\_DOCUMENT) { //文档结束

//7.读取一行数据之后 就存储到result变量中

if (eventType==XmlPullParser.START\_TAG

&&pullParser.getName().equals("string")) {

result+=pullParser.nextText()+"\n";

}

//6.获取下一个类型，进行循环判断

eventType=pullParser.next();

}

mWeatherTv.setText(result);

}

下面是要解析的文件内容

<ArrayOfString xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://WebXml.com.cn/">

<string>北京</string>

<string>2015-9-7 15:22:46</string>

<string>17℃/29℃</string>

<string>无持续风向微风</string>

<string>今日天气实况：气温：28℃；风向/风力：西南风 2级；湿度：44%；空气质量：暂无；紫外线强度：强</string>

</ArrayOfString>

## sqllite数据库

概念

创建数据库：

具有读写权限

生成目录：位于data/data/应用程序包名/databases文件目录

SQLite中创建表

SQLite中所有数据类型默认存储都是字符串类型的，没有长度限制的。超过表定义时的数据长度，也不会报错。

在Android中SQLite会自动创建一张名为android\_metadata的表。用于存储当前的语言环境的。在SQLite中id推荐使用\_id。

SQLite的数据库版本只能增大，不能减小。

游标：在访问数据库中表结构时，想访问表中的某一行的时候，数据库内部有一个快速的定位方式，这个定位方式是通过索引来实现的。游标相当于数组的指针，通过游标的上下移动来查找数据

创建数据表步骤： 本身已内置了sqlite，不用导包之类操作

第一步 创建数据库和表：继承抽象类 SQLiteOpenHelper实现两个方法

1、继承SQLiteOpenHelper抽象类(此类没有无参构造方法)和需实现它的两个抽象方法

public class DbOpenHelper extends SQLiteOpenHelper {

百老汇老师建议优化本类的构造方法：即父类的有些构造参数在本类中定义：版本、数据库文件名等

public static final String DB\_NAME="wwl.db"; 数据库文件名

public static final int DB\_VERSION=2; 数据库版本

//抽取出数据库的表名和字段(需要在dao类中用到：看第三步)

public static final String DB\_TABLENAME="AAA"; //表名

public static final String COLUMN\_ID="\_id"; //字段

public static final String COLUMN\_NAME="name"; //字段

public static final String COLUMN\_PHONE="phone"; //字段

public static final String COLUMN\_MONEY="money"; //字段

2、本类构造方法优化后只剩上下文参数

public DbOpenHelper(Context context) {

   \* 参数一：context 上下文

    \* 参数二：mname  数据库文件的名称

    \*参数三：factory 用来创建游标对象，null就用默认的游标工厂

    \* 参数四：version 数据库的版本号，从1开始。用来更新数据库。

    \* 数据库后缀名不是固定的，写为.db只是为了方便查看和识别

super(context, DB\_NAME, null, DB\_VERSION);

}

3、 onCreate方法是自动调用的：主要初始化创建数据库表操作

安卓目录下：data/data/应用程序包名/databases/名称.db：数据库文件没有才会调用

public void onCreate(SQLiteDatabase db) {

// db.execSQL方法：执行创建表的SQL语句

db.execSQL("create table "+DB\_TABLENAME+"("+COLUMN\_ID+" integer primary key autoincrement,"

+COLUMN\_NAME+" varchar(20),"+COLUMN\_PHONE+" varchar(20))");

}

4、 onUpgrade方法是自动调用的： 升级数据库 可以在该方法里面删除/添加/修改数据库表

1、数据库版本更新时自动执行,即数据库版本号有变化的时候执行

2、int oldVersion, int newVersion用来完成跨版本升级数据库时保持数据

3、如果是垄断性行业，数据是一次有效的情况下，可以强制以前数据失效的方式升级，让用户重新输入数据

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

//新版本需要向表中添加一个字段money

db.execSQL("alter table "+DB\_TABLENAME+" add "+COLUMN\_MONEY+" varchar(10)");

}

第二步 调用操作

//1、执行该行代码，数据库不会被创建:创建了一个数据库对象

DbOpenHelper db =new DbOpenHelper(this); //此类中已经进行了数据库和表的创建

//2、执行下面语句才会创建数据库：位于data/data/应用程序包名/databases/文件夹下面

SQLiteDatabase sd = db.getWritableDatabase();

//3、创建了数据库和表后 进行插入操作

        String sql = "insert into info(name,phone)values(?,?)";

//4、在SQLite中，也可以使用占位符来传递参数

        db.execSQL(sql, new Object[]{"lisi","453231"});

 //5、释放资源

        db.close();

}

第三步 优化： 上面两步是简单操作 ;包装第一步的类进行增删改查的操作(CRUD)

优化1、定义成员变量第一步的类，通过构造方法来初始化DbOpenHelper类

public class ContactDao {

//创建上面第一步写的类:通过构造器进行调用

private DbOpenHelper mHelper;

public ContactDao(Context c) {

mHelper=new DbOpenHelper(c);

}

插入操作

public boolean insertContact(String username,String phone){

//第一次会创建或者打开数据库，该数据库具有读写权限，位于data/data/应用程序包名/databases/文件夹下面

SQLiteDatabase db = mHelper.getWritableDatabase();

//ContentValues 键值对 源代码里面只是简单的维护了一个HashMap

ContentValues values=new ContentValues();

//key 插入的数据库表的字段名 value就是要插入的值

values.put(DbOpenHelper.\_USERNAME, username);

values.put(DbOpenHelper.\_PHONE, phone);

//参数：第一个表名，第二个是表示nullColumnHack表示是否为空，第三个values表示键值对

//nullColumnHack返回值String 比如写了字段”username” 那么表字段username就不能为null

//如果nullColumnHack设置为null 所有的字段都可以为空

long rowId = db.insert(DbOpenHelper.TABLE\_NAME, null , values);

// 返回值rowId 如果是-1则说明插入失败

return rowId != -1;

}

修改操作

public boolean updateContact(String username,String newPhone){

SQLiteDatabase db = mHelper.getWritableDatabase();

ContentValues values=new ContentValues(); //设置键值对象

values.put(DbOpenHelper.\_PHONE, newPhone); //放入字段名和值

//参数1：表名，参数2：表示更新的键值对，参数3：where后面语句，参数4：where语句绑定的值

int updateRows = db.update(DbOpenHelper.TABLE\_NAME,

values, DbOpenHelper.\_USERNAME+"=?", new String[]{username});

//返回值 updateRows更新了几行就返回多少

return updateRows>0;

}

删除操作

public boolean deleteContact(String username){

SQLiteDatabase db = mHelper.getWritableDatabase();

//参数1：表名，参数2：where后面语句，参数3：where语句绑定的值

int deleteRows = db.delete(DbOpenHelper.TABLE\_NAME, "username=?", new String[]{username});

return deleteRows>0;

}

查询操作

public String queryContact(String phone){

SQLiteDatabase db = dbOpenHelper.getReadableDatabase();

//参数1：table 表示表名

//参数2：columns 返回多少列 比如 new String[]{'列名','列名2'} 行数

//参数3：selection where语句

/参数4：selectionArgs表示绑定的值

Cursor cursor = db.query(DbOpenHelper.TABLE\_NAME,new String[]{DbOpenHelper.USERNAME,DbOpenHelper.PHONE} ,

""+dbOpenHelper.PHONE+"=?", new String[]{phone}, null, null, null);

String result="";

while(cursor.moveToNext()){

//根据列名返回索引

int usernameIndex=cursor.getColumnIndex(DbOpenHelper.USERNAME);

String username = cursor.getString(usernameIndex);

int phoneIndex = cursor.getColumnIndex(DbOpenHelper.PHONE);

String phones = cursor.getString(phoneIndex);

result+=("名字 : "+username+" 电话 : "+phones+" \n");

}

return result;

}

注意：在这里有   SQLiteOpenHelper对象.getWritableDatabase();或getReadableDatabase();两种的区别在

1.getWritableDatabase();用于增删改

        --用于写数据比如你要插入一条数据插入进去是写数据    写数据就要使用getWritableDatabase();方法返回一个SQLiteDatabase对象

2.getReadableDatabase();用于查询

            --用于读数据比如你要查看一条数据就是读在数据表中读取数据进来    写数据就要使用getReadableDatabase();方法返回一个SQLiteDatabase对象

## 网络

### 获取服务器数据

网络请求通用步骤

1.创建URL地址-->String urlPath

URL url=new URL(urlPath);

2.打开一个网络链接

HttpUrlConnection conn=url.openConnection();

3.判断数据返回的状态码(响应码)

int resCode=conn.getResoponseCode();

4.判断如果resCode等于200 说明资源找到了 需要将资源读取进手机

InputStream is=conn.getInputStream();

注意:

1.使用网络权限 AndroidMenifest.xml文件配置

2.使用子线程做网络请求

3.控件的修改应该返回主线程中 runOnUIThread()

子线程更新UI步骤

1、使用Handler

跳到主线程中修改界面

1.在子线程中发送一条消息告诉主线程 想修改图片控件显示

// Message有两个参数 arg1 arg2 都是int类型 如果你想传递一个int类型数据 就可以使用它们

// Message 还提供了另外一个属性msg.obj 可以携带任何对象

final Bitmap bmp = BitmapFactory.decodeFile(file.getAbsolutePath());

Message msg=new Message(); //创建Message对象

msg.obj=bmp; //使用obj方法保存bmp（图片信息用于更改ImageView控件）

mHandler.sendMessage(msg);

2.在主线程中创建了一个Handler对象：重写方法handleMessage

private Handler mHandler=new Handler(){

//3.会自动调用下面方法：在主线程中处理消息

public void handleMessage(Message msg) {

mImageView.setImageBitmap((Bitmap) msg.obj);//获取bmp并使用ImageView展示图片

Log.v("520it", "handleMessage "+Thread.currentThread().getName());

}

};

2、使用runOnUiThread直接切换到主线程

runOnUiThread(new Runnable() {

public void run() {

mImageView.setImageBitmap(bmp);

}

});

#### 显示图片

初始：不缓存图片数据从网络获取进行展示图片

第一 布局Xml

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<ImageView

android:id="@+id/iv"

android:layout\_width="match\_parent"

android:layout\_height="0dp"

android:layout\_weight="1"/>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:onClick="upImage" //点击事件

android:layout\_weight="1"

android:text="上一张" />

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_weight="1"

android:onClick="nextImage" //点击事件

android:text="下一张" />

</LinearLayout>

</LinearLayout>

MainActivity主方法

注意事项：1、需要网络权限在AndroidManifest写：

<uses-permission android:name="android.permission.INTERNET"/>

2、主线程用来改变控件的状态的，控件状态一旦改变比如添加内容都不可以在子线程操作：

使用到的类：runOnUiThread(new Runnable() {

public void run() {

iv.setImageBitmap(bitmap);//更改了控件必须在主线程用

}

3、网络连接需要在子线程操作，高版本不允许在主线程连接，因有延迟现象导致问题

下面是正式内容：

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

1、开始运行的方法：网络中获取图片的URl地址的文本，然后把每行的图片地址保存到ArrayList集合中操作

iv = (ImageView) findViewById(R.id.iv); //获取控件ImagView

new Thread() { // 开启子线程用于网络连接线程问题：高版本后都不允许在主线程中获取网络连接

public void run() {

try {

list = new ArrayList<String>(); //1、创建arrayList集合

URL url= new URL("http://192.168.0.101:8080/img/photo.txt");//2、创建Url获取地址

HttpURLConnection conn = (HttpURLConnection) url.openConnection();//3、打开连接：强转为HttpURLConnection

int code = conn.getResponseCode(); //4、获取状态码

if(code==200) { //5、判断状态码是否成功

InputStream is = conn.getInputStream();//6、使用连接获取流

BufferedReader br =new BufferedReader(new InputStreamReader(is));//7、获取文本5行使用字符流一行一行读写

String str=null;

while((str=br.readLine()) !=null) {

list.add(str); //8、添加到集合中

}

}

extracted(i); //先执行第一个图片

} catch (Exception e) {

e.printStackTrace();

}

}.start();

}

2、extracted方法：上面步骤获取到了5个图片的URl地址，使用上面1-5相同的步骤然后把流中的图片放到ImagView中展示

private void extracted(int i){

try {

URL url = new URL(list.get(i));

HttpURLConnection conn= (HttpURLConnection) url.openConnection();

int code = conn.getResponseCode();

if(code==200) {

InputStream is = conn.getInputStream();

final Bitmap bitmap = BitmapFactory.decodeStream(is);

runOnUiThread(new Runnable() {//更改了控件必须在主线程用

public void run() {

iv.setImageBitmap(bitmap);

}

});

}

} catch (Exception e) {

e.printStackTrace();

}

}

public void upImage(View v) { //上一张单击事件方法：注意需要开辟线程进行网络请求

i--;

if(i<0) {

i=list.size()-1;

}

new Thread() {

public void run() {

extracted(i); //调用方法2

}

}.start();

}

public void nextImage(View v) { //上一张单击事件方法：注意需要开辟线程进行网络请求

i++;

if(i>list.size()-1) {

i=0;

}

new Thread() {

public void run() {

extracted(i);

}

}.start();

}

性能优化：网络上获取的图片进行缓存

优化方法extracted即可：网络读取到数据进行保存到files文件夹中，判断是否有文件

private void extracted(int i){

try {优化缓存中files文件中找图片：红色部分判断files文件夹是否有缓存的文件

String imageUrl=list.get(i); 1.拿到的图片Url地址

String string = imageUrl.substring(imageUrl.lastIndexOf("/")+1); 2.拿到图片名

File file=new File(getFilesDir(), string); 3.拿到图片存储时的全路径 判断文件是否存在

if(file.exists() && file.length()>0) { 4.如果当前的图片被缓存了 就不需要在执行网络请求了

final Bitmap bm = BitmapFactory.decodeFile(file.getAbsolutePath());

runOnUiThread(new Runnable() { 5、因为还是在子线程操作了，还需转换为主线程

public void run() {

iv.setImageBitmap(bm); //6.文件存在了 还要从文件里面读取图片

}

});

return;

}

URL url = new URL(imageUrl);

HttpURLConnection conn= (HttpURLConnection) url.openConnection();

int code = conn.getResponseCode();

if(code==200) {

InputStream is = conn.getInputStream();

final Bitmap bitmap = BitmapFactory.decodeStream(is);

优化缓存图片：红色部分是把图片缓存到files文件中，安卓提供了方便的方法bitmap.compress()

String imageName=imageUrl.substring(imageUrl.lastIndexOf("/")+1);

//参数一：文件类型，参数二：图片从流转成文件保存质量取值：0~100，参数三：保存位置

bitmap.compress(CompressFormat.PNG, 100, openFileOutput(imageName, MODE\_PRIVATE));

runOnUiThread(new Runnable() {

public void run() {

iv.setImageBitmap(bitmap);//更改了控件必须在主线程用

}

});

} catch (Exception e) {

e.printStackTrace();

}

#### 新闻客户端

1. 自定义控件MyImageView
2. 主布局：xml activity\_main使用listView控件
3. 网络中获取的是Xml内容：使用XmlPullParser解析并封装到自定义Bean中
4. 辅布局：xinwen\_layout.xml使用RelativeLayout：有自定义图片控件和3个TextView

第一步 自定义控件ImageView

请看[自定义控件](#_自定义控件)：有xml布局内容和类的定义

第二步 Util的代码

public class Util {

/\*\*

\* 获取网络资源xml操作

\* @param spec

\* @return 网络异常有可能返回null

\*/

public static InputStream UrlResource(String spec) {

//第一步创建Url

URL url;

InputStream inputStream=null ;

try {

url = new URL(spec);

//第二步打开连接

HttpURLConnection httpUrl = (HttpURLConnection) url.openConnection();

//第三步获取服务器返回来的状态码

int code = httpUrl.getResponseCode();

//4、判断状态码

if(code==200) {

inputStream = httpUrl.getInputStream();//获取流

}

} catch (MalformedURLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return inputStream;

}

public static ArrayList<XmlBean> pullXml(InputStream is){

ArrayList<XmlBean> list =new ArrayList<XmlBean>();

XmlPullParser xpp= Xml.newPullParser();

try {

if(is !=null) {

xpp.setInput(is, "utf-8");

int type = xpp.getEventType(); //获取当前状态码

XmlBean xmlBean=null;

while(type !=XmlPullParser.END\_DOCUMENT) { //不等于结束的文档状态

if(type==XmlPullParser.START\_TAG) { //获取开标签

if("item".equals(xpp.getName())) {

xmlBean=new XmlBean();

}else if("title".equals(xpp.getName())) {

String title= xpp.nextText(); //获取开标签后面的文本值：<a>你好，你好的值

xmlBean.setTitle(title);

}else if("description".equals(xpp.getName())) {

String description=xpp.nextText(); //获取开标签后面的文本值：<a>你好，你好的值

xmlBean.setDescription(description);

}else if("image".equals(xpp.getName())) {

String image=xpp.nextText(); //获取开标签后面的文本值：<a>你好，你好的值

xmlBean.setImage(image);

}else if("type".equals(xpp.getName())) {

String type1=xpp.nextText(); //获取开标签后面的文本值：<a>你好，你好的值

xmlBean.setType(Integer.parseInt(type1));

}else if("comment".equals(xpp.getName())) {

String comment=xpp.nextText(); //获取开标签后面的文本值：<a>你好，你好的值

xmlBean.setComment(comment);

}

}

if(type==XmlPullParser.END\_TAG) { //闭标签

if("item".equals(xpp.getName())) {

list.add(xmlBean);

}

}

type= xpp.next();//获取下一个状态码

}

System.out.println("");

return list;

}

} catch (XmlPullParserException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return null;

}

}

第三步 主代码和MyAdapter中的代码

MainActivity代码

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

//获取ListView

ListView lv = (ListView) findViewById(R.id.lv);

adapter= new MyAdapter();

lv.setAdapter(adapter);

new Thread() {

public void run() {

//获取内容

ArrayList<XmlBean> list = Util.pullXml(Util.UrlResource("http://192.168.0.100:8080/news.xml"));

adapter.setDatas(list);

}

}.start();

}

}

MyAdapter代码

public class MyAdapter extends BaseAdapter {

private ArrayList<XmlBean> list;

public void setDatas(ArrayList<XmlBean> list) {

this.list=list;

}

@Override

public int getCount() {

return list==null?0:list.size();

}

class HelpView{

MyImageView iv;

TextView titleTV;

TextView descriptionTV;

TextView commentTV;

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

HelpView hv =null;

if(convertView==null) {

hv = new HelpView();

LayoutInflater inflater = LayoutInflater.from(parent.getContext());

convertView = inflater.inflate(R.layout.xinwen\_layout, null);

hv.iv = (MyImageView) convertView.findViewById(R.id.iv);使用自定义的图片控件

hv.titleTV=(TextView) convertView.findViewById(R.id.titleTV);

hv.descriptionTV=(TextView) convertView.findViewById(R.id.descriptionTV);

hv.commentTV=(TextView) convertView.findViewById(R.id.commentTV);

convertView.setTag(hv);

}else {

convertView=(View) convertView.getTag();

}

XmlBean bean = list.get(position);

hv.titleTV.setText(bean.getTitle());

hv.descriptionTV.setText(bean.getDescription());

if(bean.getType()==1) {

hv.commentTV.setText("评论："+bean.getComment());

}else if(bean.getType()==2) {

hv.commentTV.setText("专题");

}else if(bean.getType()==3) {

hv.commentTV.setText("主题");

}

hv.iv.setImageUrl(bean.getImage());使用自定义的图片控件自己定义的方法

return convertView;

}

### 发送数据服务器

#### JDK原始

GET请求：进行发送数据至服务器

其实和获取服务器数据几乎一致,只多了一行代码：conn.setRequestMethod("GET");

例子：登录操作：把账号密码：www和123456，发送给服务器则打印成功，否则则失败

public void onClickB(View v){

new Thread() {

public void run() {

try {

String username = et1.getText().toString();

String password = et2.getText().toString();

String urlPath="http://192.168.0.100:8080/wwl/ls?userName="+username+"&password="+password;

URL url= new URL(urlPath); 1.获取客户端(浏览器)

HttpURLConnection conn = (HttpURLConnection) url.openConnection(); 2、打开Http连接

conn.setRequestMethod("GET");3、地址有参数：需发送数据至服务器 必须大写

int code = conn.getResponseCode();4、获取返回的状态码

if(code==200) {5、判断

InputStream is = conn.getInputStream();6、获取数据流

BufferedReader bf = new BufferedReader(new InputStreamReader(is));

final String line = bf.readLine();7、只有一行数据

runOnUiThread(new Runnable() {

public void run() {

Toast.makeText(MainActivity.this, line, 0).show(); //7、打印数据

}

});

}

}catch(Exception e) {

e.printStackTrace();

}

}

}.start();

}

Post请求和get区别：多了两个请求头：Content-Type和Content-Length

很多一样这里只写重点地方

new Thread() {

public void run() {

try {

String username = et1.getText().toString();

String password = et2.getText().toString();

Log.v("wwl", username);

Log.v("wwl", password);

String urlPath="http://192.168.0.100:8080/wwl/ls";

URL url= new URL(urlPath);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");

//和get区别 ：查看请求头多了两个请求头：Content-Type和Content-Length，需设置

String params="userName="+username+"&password="+password; //post请求datafrom中参数内容

conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");//post有普通，和上传图片

conn.setRequestProperty("Content-Length", params.length()+"");//设置请求头：参数内容长度

//\*\*\*\*\*向服务器提交数据OutputStream：安卓下HttpURLConnection的输出流开关默认是关闭的

conn.setDoOutput(true);

conn.getOutputStream().write(params.getBytes());

……..下面内容省略和get一模一样

#### HttpClient

安卓6.0以后已经抛弃了它

Get操作

// HttpUrlConnection方式API 面向过程

// HttpClient是一个面向对象(接口)的API

// 1.拿到两个EditText的值

final String username = mUserNameEt.getText().toString();

final String pwd = mPwdEt.getText().toString();

new Thread() {

public void run() {

try {

// 1.创建了一个浏览器

HttpClient client = new DefaultHttpClient();

// 2.声明一个Get请求

String urlPath = "http://192.168.35.10:8080/login/login?"

+ "username=" + username + "&pwd=" + pwd;

HttpGet request = new HttpGet(urlPath);

// 3.通过浏览器执行一个HttpGet请求 返回了一个HttpResponse响应对象

// HttpUriRequest 这里封装Http请求的几种请求方式：f4查看

HttpResponse response = client.execute(request);

// 4.返回结果 响应码==200

int responseCode = response.getStatusLine().getStatusCode();

if (responseCode == 200) {

// 5.获取响应的数据 HttpEntity实体 该对象封装了返回的数据

HttpEntity entity = response.getEntity();

InputStream is = entity.getContent();

BufferedReader reader = new BufferedReader(new InputStreamReader(is));

final String line = reader.readLine();

runOnUiThread(new Runnable() {

public void run() {

Toast.makeText(MainActivity.this, line, Toast.LENGTH\_SHORT).show();

}

});

}

} catch (Exception e) {

e.printStackTrace();

}

}.start();

}

Post请求

//1.创建一个浏览器

HttpClient client=new DefaultHttpClient();

//2.声明一个HttpPost请求

String urlPath="http://192.168.35.10:8080/loginPost/login";

HttpPost request=new HttpPost(urlPath);

//3.Post是将数据添加到Form表单中 思考:怎么将数据添加到请求里面

//不管是请求的数据还是返回来的数据 所有的数据都封装在HttpEntity里面

//Pair 双BasicNameValuePair键值对

ArrayList<BasicNameValuePair> parameters=new ArrayList<BasicNameValuePair>();

parameters.add(new BasicNameValuePair("username", username));

parameters.add(new BasicNameValuePair("pwd", pwd));

HttpEntity entity=new UrlEncodedFormEntity(parameters, "utf-8");

request.setEntity(entity);

//4.执行一个HttpPost请求

HttpResponse response = client.execute(request);

//5.判断响应码

int responseCode = response.getStatusLine().getStatusCode();

if (responseCode==200) {

//6.读取服务器里面的数据

HttpEntity responseEntity = response.getEntity();

InputStream is = responseEntity.getContent();

BufferedReader reader = new BufferedReader(new InputStreamReader(is));

final String line = reader.readLine();

省略……..

##### 基于HttpClient开源框架

还是JDK原始中的例子要求

第一步 导源码(不是导包)名字: android-async-http-master

进入android-async-http-master\sample\src\main\java目录把com文件夹放到自己工程中

第二步 代码

final String username = mUserNameEt.getText().toString();

final String pwd = mPwdEt.getText().toString();

//如下的操作自带了子线程

//1初始化了一个浏览器

AsyncHttpClient client=new AsyncHttpClient();

//2准备网址

String urlPath="http://192.168.35.10:8080/loginPost/login";

//3、准备params RequestParams请求的参数

RequestParams params=new RequestParams();

params.put("username", username);

params.put("pwd", pwd);

// 4、执行Post请求ResponseHandlerInterface的子类事件实现两个方法：一个200成功和失败的

client.post(urlPath, params, new TextHttpResponseHandler() {

//响应码200该方法下的代码 是在主线程中运行

//responseString 就是服务器返回的数据

public void onSuccess(int statusCode, Header[] headers, String responseString) {

Toast.makeText(MainActivity.this, responseString, 0).show();

}

//响应码 404 500 505方法

public void onFailure(int statusCode, Header[] headers,

String responseString, Throwable throwable) {

}

});

}

文件上传操作

AsyncHttpClient client=new AsyncHttpClient();

String urlPath="http://192.168.35.10:8080/upload/upload";

RequestParams params=new RequestParams();

params.put("fName", "phone.png"); //查看了浏览器的上传状态名字必须：fName

File file=new File(getFilesDir(),"aa.png");

params.put("file", file); //文件源必须file

client.post(urlPath, params, new TextHttpResponseHandler() {

public void onSuccess(int statusCode, Header[] headers, String responseString) {

Toast.makeText(MainActivity.this, responseString, 0).show();

}

public void onFailure(int statusCode, Header[] headers,

String responseString, Throwable throwable) {

}

});

}